5.2 소스코드

1. DBConnect.java: MySQL 연동

```
public class DBConnect {
	private static Connection conn = null;

	public static Connection getConnection() {
		if (conn == null) {
			try {
				String driver = "com.mysql.cj.jdbc.Driver";
				String url =
"jdbc:mysql://localhost:3306/shopmallmanageDB?useUnicode=true&characterEncoding=utf8";
				String user = "root";
				String pwd = "1234";
				Class.forName(driver);
				conn = DriverManager.getConnection(url, user, pwd);
				System.out.println("Connect");
			} catch (ClassNotFoundException e) {
				e.printStackTrace();
			} catch (SQLException e) {
				e.printStackTrace();
			}
		}
		return conn;
	}
}
```

2. Frame.java: 프레임 구성

```
public class Frame {
	private static JFrame frame;
	private static List<JPanel> pList = setPanel();

	public static void main(String[] args) {
		frame = new JFrame("쇼핑몰 관리 프로그램");
		frame.setVisible(true);
		frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
		frame.setResizable(false);
		frame.setSize(1200, 800);
		frame.setPreferredSize(new Dimension(1200, 800));
		frame.setLocationRelativeTo(null);
		frame.getContentPane().setLayout(null);

		addPanel();
	}
	//화면을 구성할 패널들 생성
	private static ArrayList<JPanel> setPanel() {
```

```
                ArrayList<JPanel> temp = new ArrayList<>();
                temp.add(HomePanel.getPanel());
                temp.add(MenuPanel.getPanel());
                temp.add(MemberPanel.getPanel());
                temp.add(OrderPanel.getPanel());
                temp.add(ProductPanel.getPanel());
                return temp;
        }
        //화면 패널을 프레임에 추가
        private static void addPanel() {
                for (JPanel panel : pList) {
                        frame.getContentPane().add(panel);
                }
        }
}
```

3. HomePanel.java: 메인화면 및 기능구현

3-1 화면 UI 기능

```
   //화면 전환시 패널의 가시성 설정
   public static void initVisible(int n) {
      boolean vHome = false;
      boolean vMenu = false;
      boolean vMember = false;
      boolean vOrder = false;
      boolean vProduct = false;
      switch (n) {
      case 1:
         vHome = true;
         vMenu = false;
         vMember = false;
         vOrder = false;
         vProduct = false;
         break;
      case 2:
         vHome = false;
         vMenu = true;
         vMember = true;
         vOrder = false;
         vProduct = false;
         break;
      case 3:
         vHome = false;
         vMenu = true;
         vMember = false;
         vOrder = true;
         vProduct = false;
```

```
            break;
        case 4:
            vHome = false;
            vMenu = true;
            vMember = false;
            vOrder = false;
            vProduct = true;
        }

        home.setVisible(vHome);
        menu.setVisible(vMenu);
        member.setVisible(vMember);
        order.setVisible(vOrder);
        product.setVisible(vProduct);
    }
}
```

3-2 초기화면에서 회원수 표시 기능

```
    public static void setMemberNo() {
        String sql = "SELECT count(id) FROM membertbl";
        String r = "";
        try {
            Statement stmt = conn.createStatement();
            ResultSet rs = stmt.executeQuery(sql);
            rs.next();
            r = rs.getString(1);
            if (rs != null)
                rs.close();
            if (stmt != null)
                stmt.close();
        } catch (SQLException e) {
            e.printStackTrace();
        }
        memNoTextField.setText(r + "명");
        if (memNoTextField != null)
            memNoTextField.repaint();
    }
```

3-3 초기화면에서 이달의 판매금액 표시 기능

```
    public static void setSales() {
        String sql = "select sum(ordertbl.totalprice) from ordertbl where substr(orderdate,6,7)
= ?";
        int r = 0;
        String str = "";
        String month = dateTextField.getText().substring(6, 8);
        try {
```

```
            PreparedStatement pstmt = conn.prepareStatement(sql);
            pstmt.setInt(1, Integer.parseInt(month));
            ResultSet rs = pstmt.executeQuery();
            rs.next();
            r = rs.getInt(1);
            DecimalFormat formatter = new DecimalFormat("###,###,###,###");
            str = formatter.format((Number) r);
            if (rs != null)
                rs.close();
            if (pstmt != null)
                pstmt.close();
        } catch (SQLException e) {
            e.printStackTrace();
        }
        textFieldSales.setText(str + "원");
        if (textFieldSales != null)
            textFieldSales.repaint();

    }
```

4. MenuPanel.java: 메뉴버튼 화면 및 기능구현

```
public class MenuPanel {
        private static JPanel menu = null;

        public MenuPanel() {
                getPanel();
        }

        // 패널들 생성하고 메소드를 호출시 패널 반환
        public static JPanel getPanel() {
                if (menu == null) {
                        menu = new JPanel();
                        menu.setBackground(new Color(255, 0, 0, 0));
                        menu.setBounds(0, 0, 1185, 100);
                        menu.setLayout(null);

                        JPanel panel = new JPanel();
                        panel.setBackground(UIManager.getColor("Button.disabledShadow"));
                        panel.setBounds(307, 0, 571, 100);
                        menu.add(panel);
                        panel.setLayout(null);

                        RoundedButton btnHome = new RoundedButton("Home");
                        btnHome.setBounds(21, 12, 116, 76);
```

```
                        panel.add(btnHome);
                        btnHome.setText("\uCC98\uC74C");
                        btnHome.setForeground(SystemColor.text);
                        btnHome.setBackground(new Color(135, 206, 250));
                        btnHome.setFont(new Font("맑은 고딕", Font.PLAIN, 30));
                        btnHome.setPreferredSize(new Dimension(116, 63));
                        btnHome.addActionListener(new ActionListener() {
                                public void actionPerformed(ActionEvent e) {
                                        HomePanel.initVisible(1);
                                        HomePanel.setMemberNo();
                                        HomePanel.setSales();
                                }
                        });

                        RoundedButton btnMember = new RoundedButton("\uBA64\uBC84");
                        btnMember.setToolTipText(
                                        "<html>입력 칸에 입력값, 입력값  형식으로 입력시 여러 개의 값 검색<br>이름 검색할 때 성씨 검색 방법 김__, 이__, 박__ 과 같이 언더바 2개 사용하여 검색</html>");
                        btnMember.setText("\uD68C\uC6D0");
                        btnMember.setBounds(158, 12, 116, 76);
                        panel.add(btnMember);

                        btnMember.setForeground(SystemColor.text);
                        btnMember.setBackground(new Color(135, 206, 250));
                        btnMember.setFont(new Font("맑은 고딕", Font.PLAIN, 30));
                        btnMember.setPreferredSize(new Dimension(116, 63));

                        btnMember.addActionListener(new ActionListener() {
                                public void actionPerformed(ActionEvent e) {
                                        HomePanel.initVisible(2);
                                        MemberPanel.getTable();
                                }
                        });

                        RoundedButton btnOrder = new RoundedButton("\uC8FC\uBB38");
                        btnOrder.setToolTipText("입력 칸에 입력값, 입력값  형식으로 입력시 여러 개의 값 검색");
                        btnOrder.setBounds(295, 12, 116, 76);
                        panel.add(btnOrder);
                        btnOrder.setForeground(SystemColor.text);
                        btnOrder.setBackground(new Color(135, 206, 250));
                        btnOrder.setFont(new Font("맑은 고딕", Font.PLAIN, 30));
                        btnOrder.setPreferredSize(new Dimension(116, 63));
                        btnOrder.addActionListener(new ActionListener() {
```

```
                                public void actionPerformed(ActionEvent e) {
                                        HomePanel.initVisible(3);
                                        OrderPanel.getTable();
                                }
                        });

                        RoundedButton btnProduct = new RoundedButton("\uC0C1\uD488");
                        btnProduct.setToolTipText("<html>" + "상품명 칸에 상품명 입력시 상품
명 중복없이 출력<br>" + "분류 칸에 분류 입력시 분류 중복없이 출력<br>"
                                        + "입력 칸에 입력값, 입력값  형식으로 입력시 여러 개의
값 검색<br>" + "</html>");
                        btnProduct.setBounds(432, 12, 116, 76);
                        panel.add(btnProduct);
                        btnProduct.setForeground(SystemColor.text);
                        btnProduct.setBackground(new Color(135, 206, 250));
                        btnProduct.setFont(new Font("맑은 고딕", Font.PLAIN, 30));
                        btnProduct.setPreferredSize(new Dimension(116, 63));
                        btnProduct.addActionListener(new ActionListener() {
                                public void actionPerformed(ActionEvent e) {
                                        HomePanel.initVisible(4);
                                        ProductPanel.getTable();
                                }
                        });
                        BufferedImage image = null;
                        try {
                                image = ImageIO.read(new File("./source/1.jpg"));
                        } catch (IOException e1) {
                                e1.printStackTrace();
                        }
                        JLabel imageLabel = new JLabel(new ImageIcon(image));
                        imageLabel.setBounds(0, 0, 1184, 761);
                        imageLabel.setPreferredSize(new Dimension(1184, 661));
                        menu.add(imageLabel);

                }
                return menu;
        }
}
```

5. MemberPanel.java: 회원화면 및 기능구현
5-1 회원 정보 입력 기능
// 회원 정보를 입력하는 기능 private static void insertMember() {

```
                String sql = "INSERT INTO membertbl values(?,?,?,?)";
                try {
                        PreparedStatement pstmt = conn.prepareStatement(sql);
                        pstmt.setString(1, textID.getText());
                        pstmt.setString(2, textPhone.getText());
                        pstmt.setString(3, textAddress.getText());
                        pstmt.setString(4, textName.getText());
                        pstmt.executeUpdate();
                        model.setRowCount(0);
                        getTable();

                        if (pstmt != null)
                                pstmt.close();
                        String error = "회원 정보를 추가했습니다.";
                        JLabel lblError = new JLabel(error);
                        lblError.setFont(new Font("맑은 고딕", Font.PLAIN, 15));
                        JOptionPane.showMessageDialog(null, lblError, "Successful",
JOptionPane.PLAIN_MESSAGE);
                } catch (SQLException e1) {
                        String error = "상품 정보를 추가하지 못했습니다.";
                        JLabel lblError = new JLabel(error);
                        lblError.setFont(new Font("맑은 고딕", Font.PLAIN, 15));
                        JOptionPane.showMessageDialog(null, lblError, "Error",
JOptionPane.PLAIN_MESSAGE);
                }
        }
```

5-2 회원 정보 갱신 기능

```
// 회원 정보를 갱신하는 기능
        private static void updateMember() {
                String sql = "UPDATE membertbl SET id = ?, name = ?, phonenum = ?,
address = ? where id = ?";
                try {
                        PreparedStatement pstmt = conn.prepareStatement(sql);
                        pstmt.setString(1, textID.getText());
                        pstmt.setString(2, textName.getText());
                        pstmt.setString(3, textPhone.getText());
                        pstmt.setString(4, textAddress.getText());
                        pstmt.setString(5, (String) model.getValueAt(select, 0));
                        pstmt.executeUpdate();
                        model.setRowCount(0);
                        getTable();

                        if (pstmt != null)
```

```
                        pstmt.close();
                    String error = "회원 정보를 갱신했습니다.";
                    JLabel lblError = new JLabel(error);
                    lblError.setFont(new Font("맑은 고딕", Font.PLAIN, 15));
                    JOptionPane.showMessageDialog(null, lblError, "Successful",
JOptionPane.PLAIN_MESSAGE);
                } catch (Exception e1) {
                    String error = "회원 정보를 갱신 하지 못했습니다.";
                    JLabel lblError = new JLabel(error);
                    lblError.setFont(new Font("맑은 고딕", Font.PLAIN, 15));
                    JOptionPane.showMessageDialog(null, lblError, "Error",
JOptionPane.PLAIN_MESSAGE);
                }
        }

        // DB에서 데이터를 불러와 테이블로 넣는 기능
        public static void getTable() {
                model.setRowCount(0);
                try {
                    String sql = "SELECT * FROM membertbl";
                    pstmt = conn.prepareStatement(sql);
                    rs = pstmt.executeQuery();
                    while (rs.next()) {
                        model.addRow(new Object[] { rs.getString("ID"),
rs.getString("name"), rs.getString("phonenum"),
                                rs.getString("address") });
                    }
                    textFieldSelectCnt.setText(Integer.toString(model.getRowCount()));
                    if (rs != null)
                        rs.close();
                    if (pstmt != null)
                        pstmt.close();
                } catch (SQLException e) {
                    e.printStackTrace();
                }
        }
```

5-3 회원 정보 삭제 기능

```
// 회원 정보를 삭제하는 기능
        private static void deleteMember() {
                String sql = "DELETE FROM membertbl where id = ?";
                try {
                    PreparedStatement pstmt = conn.prepareStatement(sql);
                    pstmt.setString(1, textID.getText());
```

```
                        pstmt.executeUpdate();
                        model.setRowCount(0);
                        getTable();
                        if (pstmt != null)
                                pstmt.close();
                        String error = "회원 정보를 삭제했습니다.";
                        JLabel lblError = new JLabel(error);
                        lblError.setFont(new Font("맑은 고딕", Font.PLAIN, 15));
                        JOptionPane.showMessageDialog(null, lblError, "Successful",
JOptionPane.PLAIN_MESSAGE);
                } catch (SQLException e1) {
                        String error = "주문 내역이 있는 회원이라 삭제할 수 없습니다.";
                        JLabel lblError = new JLabel(error);
                        lblError.setFont(new Font("맑은 고딕", Font.PLAIN, 15));
                        JOptionPane.showMessageDialog(null, lblError, "Error",
JOptionPane.PLAIN_MESSAGE);
                }
        }
```

5-4 회원 정보 검색 기능

```
// 회원 정보를 검색하는 기능
        private static void searchMember(String id, String name, String phoneNum, String
address) {
                String sql = "SELECT * FROM membertbl WHERE ";
//              String sql = "SELECT * FROM membertbl WHERE id = ? and name = ? and
phonenum = ? and address = ?";
                if (!id.equals("")) {
                        st = new StringTokenizer(id, ",");
                        sql = sql + "(";
                        while (st.hasMoreTokens()) {
                                sql = sql + "id like ? or ";
                                st.nextToken();
                        }
                        sql = sql.substring(0, sql.length() - 3);
                        sql = sql + ")";
                        sql = sql + " and ";
                }
                if (!name.equals("")) {
                        st = new StringTokenizer(name, ",");
                        sql = sql + "(";

                        while (st.hasMoreTokens()) {
                                sql = sql + "name like ? or ";
                                st.nextToken();
```

```
                        }
                        sql = sql.substring(0, sql.length() - 3);
                        sql = sql + ")";
                        sql = sql + " and ";
                }
                if (!phoneNum.equals("")) {
                        st = new StringTokenizer(phoneNum, ",");
                        sql = sql + "(";
                        while (st.hasMoreTokens()) {
                                sql = sql + "phoneNum like ? or ";
                                st.nextToken();
                        }
                        sql = sql.substring(0, sql.length() - 3);
                        sql = sql + ")";
                        sql = sql + " and ";
                }
                if (!address.equals("")) {
                        st = new StringTokenizer(address, ",");
                        sql = sql + "(";
                        while (st.hasMoreTokens()) {
                                sql = sql + "address like ? or ";
                                st.nextToken();
                        }
                        sql = sql.substring(0, sql.length() - 3);
                        sql = sql + ")";
                        sql = sql + " and ";
                }

                sql = sql.substring(0, sql.length() - 4);

                try {
                        int idx = 1;
                        pstmt = conn.prepareStatement(sql);
                        if (!id.equals("")) {
                                st = new StringTokenizer(id, ",");
                                while (st.hasMoreTokens()) {
                                        pstmt.setString(idx++, "%" + st.nextToken().trim() +
"%");
                                }
                        }
                        if (!name.equals("")) {
                                st = new StringTokenizer(name, ",");
                                while (st.hasMoreTokens()) {
                                        pstmt.setString(idx++, "%" + st.nextToken().trim() +
"%");
```

```
                                        }
                                }
                                if (!phoneNum.equals("")) {
                                        st = new StringTokenizer(phoneNum, ",");
                                        while (st.hasMoreTokens()) {
                                                pstmt.setString(idx++, "%" + st.nextToken().trim() +
"%");
                                        }
                                }
                                if (!address.equals("")) {
                                        st = new StringTokenizer(address, ",");
                                        while (st.hasMoreTokens()) {
                                                pstmt.setString(idx++, "%" + st.nextToken().trim() +
"%");
                                        }
                                }
                                System.out.println(sql);
                                rs = pstmt.executeQuery();
                                model.setRowCount(0);
                                while (rs.next()) {
                                        model.addRow(new Object[] { rs.getString("ID"),
rs.getString("name"), rs.getString("phonenum"),
                                                        rs.getString("address") });
                                }
                                if (rs != null)
                                        rs.close();
                                if (pstmt != null)
                                        pstmt.close();
                        } catch (SQLException e1) {
                                e1.printStackTrace();
                        }
                }
```

6. OrderPanel.java: 주문화면 및 기능구현

6-1 주문 정보 입력 기능

```
// 주문 정보를 입력하는 기능
        private static void insertOrder() {
                try {
                        String sql = "Select amount from producttbl where productNo = ?";
                        pstmt = con.prepareStatement(sql);
                        pstmt.setInt(1, Integer.parseInt(textProductNo.getText()));
                        rs = pstmt.executeQuery();
                        rs.next();
                        if (rs.getInt("amount") >= Integer.parseInt(textOrderAmount.getText()))
{
```

```
                                sql = "INSERT INTO ordertbl(ID, productNo, orderAmount,
orderDate) VALUES(?, ?, ?, ?)";
                                pstmt = con.prepareStatement(sql);
                                pstmt.setString(1, textID.getText());
                                pstmt.setInt(2, Integer.parseInt(textProductNo.getText()));
                                pstmt.setInt(3, Integer.parseInt(textOrderAmount.getText()));
                                String date = textOrderDate.getText();
                                java.sql.Date sDate = java.sql.Date.valueOf(date);
                                pstmt.setDate(4, sDate);
                                pstmt.executeUpdate();
                                String error = "주문 정보를 추가했습니다.";
                                JLabel lblError = new JLabel(error);
                                lblError.setFont(new Font("맑은 고딕", Font.PLAIN, 15));
                                JOptionPane.showMessageDialog(null, lblError, "Successful",
JOptionPane.PLAIN_MESSAGE);
                        } else {
                                String error = "주문량이 재고보다 많습니다.";
                                JLabel lblError = new JLabel(error);
                                lblError.setFont(new Font("맑은 고딕", Font.PLAIN, 15));
                                JOptionPane.showMessageDialog(null, lblError, "Error",
JOptionPane.PLAIN_MESSAGE);
                        }
                        getTable();
                } catch (Exception ex) {
                        String error = "주문 정보를 추가하지 못했습니다.";
                        JLabel lblError = new JLabel(error);
                        lblError.setFont(new Font("맑은 고딕", Font.PLAIN, 15));
                        JOptionPane.showMessageDialog(null, lblError, "Error",
JOptionPane.PLAIN_MESSAGE);
                }
        }
```

6-2 주문 정보 갱신 기능

```
// 주문 정보를 갱신하는 기능
        private static void updateOrder() {
                try {
                        String sql = "Select amount from producttbl where productNo = ?";
                        pstmt = con.prepareStatement(sql);
                        pstmt.setInt(1, Integer.parseInt(textProductNo.getText()));
                        rs = pstmt.executeQuery();
                        rs.next();
                        if (rs.getInt("amount") >= Integer.parseInt(textOrderAmount.getText()))
{
                                sql = "UPDATE ordertbl SET ID=?, productNo=?, " + "
orderAmount=?, orderDate=? WHERE orderNo = ?";
```

```
                                pstmt = con.prepareStatement(sql);
                                pstmt.setString(1, textID.getText());
                                pstmt.setInt(2, Integer.parseInt(textProductNo.getText()));
                                pstmt.setInt(3, Integer.parseInt(textOrderAmount.getText()));
                                String date = textOrderDate.getText();
                                java.sql.Date sDate = java.sql.Date.valueOf(date);
                                pstmt.setDate(4, sDate);
                                pstmt.setInt(5, Integer.parseInt(textOrderNo.getText()));
                                pstmt.executeUpdate();
                                pstmt.close();
                                String error = "주문 정보를 갱신했습니다.";
                                JLabel lblError = new JLabel(error);
                                lblError.setFont(new Font("맑은 고딕", Font.PLAIN, 15));
                                JOptionPane.showMessageDialog(null, lblError, "Successful",
JOptionPane.PLAIN_MESSAGE);
                        } else {
                                String error = "주문 정보를 갱신하지 못했습니다.";
                                JLabel lblError = new JLabel(error);
                                lblError.setFont(new Font("맑은 고딕", Font.PLAIN, 15));
                                JOptionPane.showMessageDialog(null, lblError, "Error",
JOptionPane.PLAIN_MESSAGE);
                        }
                        getTable();
                } catch (Exception ex) {
                        String error = "주문 정보를 갱신하지 못했습니다.";
                        JLabel lblError = new JLabel(error);
                        lblError.setFont(new Font("맑은 고딕", Font.PLAIN, 15));
                        JOptionPane.showMessageDialog(null, lblError, "Error",
JOptionPane.PLAIN_MESSAGE);
                }
        }
```

6-3 주문 정보 삭제 기능

```
// 주문 정보를 삭제하는 기능
        private static void deleteOrder() {
                String sql = "DELETE FROM ordertbl WHERE orderNo = ?";
                try {
                        pstmt = con.prepareStatement(sql);
                        pstmt.setInt(1, Integer.parseInt(textOrderNo.getText()));
                        pstmt.executeUpdate();
                        getTable();
                        pstmt.close();
                        String error = "주문 정보를 삭제했습니다.";
                        JLabel lblError = new JLabel(error);
                        lblError.setFont(new Font("맑은 고딕", Font.PLAIN, 15));
```

```
                JOptionPane.showMessageDialog(null, lblError, "Successful",
JOptionPane.PLAIN_MESSAGE);
            } catch (Exception ex) {
                String error = "주문 정보를 삭제하지 못했습니다.";
                JLabel lblError = new JLabel(error);
                lblError.setFont(new Font("맑은 고딕", Font.PLAIN, 15));
                JOptionPane.showMessageDialog(null, lblError, "Error",
JOptionPane.PLAIN_MESSAGE);
            }
        }
```

6-4 주문 정보 검색 기능

```
// 주문 정보를 검색하는 기능
        private static void searchOrder(String orderNo, String ID, String productNo, String
orderAmount, String orderDate,
                String totalPrice, int comboProductNo, int comboOrderAmount, int
comboTotalPrice) {
            String sql = "SELECT * FROM ordertbl WHERE ";

            if (!orderNo.equals("")) {
                st = new StringTokenizer(orderNo, ",");
                sql = sql + "(";
                while (st.hasMoreTokens()) {
                    sql = sql + "Cast(orderNo as char(10)) like ? or ";
                    st.nextToken();
                }
                sql = sql.substring(0, sql.length() - 3);
                sql = sql + ")";
                sql = sql + " and ";
            }
            if (!ID.equals("")) {
                if (ID.toUpperCase().equals("ID")) {
                    sql = "select m.id, count(orderAmount) as countID,
sum(totalPrice) as sumID from membertbl m left outer join ordertbl o on o.id = m.id group
by id      ";
                } else {
                    st = new StringTokenizer(ID, ",");
                    sql = sql + "(";
                    while (st.hasMoreTokens()) {
                        sql = sql + "ID like ? or ";
                        st.nextToken();
                    }
                    sql = sql.substring(0, sql.length() - 3);
                    sql = sql + ")";
                    sql = sql + " and ";
```

```
                        }
                }
                if (!productNo.equals("")) {
                        if (productNo.equals("번호")) {
                                sql = "Select p.productNo, count(orderAmount) as countPN, sum(totalPrice) as sumPN from producttbl p left outer join ordertbl o on p.productNo = o.productNo group by productNo      ";
                        } else {
                                st = new StringTokenizer(productNo, ",");
                                sql = sql + "(";
                                switch (comboProductNo) {
                                case (0):
                                        while (st.hasMoreTokens()) {
                                                sql = sql + "productNo = ? or ";
                                                st.nextToken();
                                        }
                                        break;
                                case (1):
                                        while (st.hasMoreTokens()) {
                                                sql = sql + "Cast(productNo as char(10)) like ? or ";

                                                st.nextToken();
                                        }
                                        break;
                                case (2):
                                        while (st.hasMoreTokens()) {
                                                sql = sql + "productNo >= ? or ";
                                                st.nextToken();
                                        }
                                        break;
                                case (3):
                                        while (st.hasMoreTokens()) {
                                                sql = sql + "productNo <= ? or ";
                                                st.nextToken();
                                        }
                                        break;
                                }
                                sql = sql.substring(0, sql.length() - 3);
                                sql = sql + ")";
                                sql = sql + " and ";
                        }
                }
                if (!orderAmount.equals("")) {
                        st = new StringTokenizer(orderAmount, ",");
                        sql = sql + "(";
```

```
                        switch (comboOrderAmount) {
                        case (0):
                                while (st.hasMoreTokens()) {
                                        sql = sql + "orderAmount = ? or ";
                                        st.nextToken();
                                }
                                break;
                        case (1):
                                while (st.hasMoreTokens()) {
                                        sql = sql + "Cast(orderAmount as char(10)) like ? or
";

                                        st.nextToken();
                                }
                                break;
                        case (2):
                                while (st.hasMoreTokens()) {
                                        sql = sql + "orderAmount >= ? or ";
                                        st.nextToken();
                                }
                                break;
                        case (3):
                                while (st.hasMoreTokens()) {
                                        sql = sql + "orderAmount <= ? or ";
                                        st.nextToken();
                                }
                                break;
                        }
                        sql = sql.substring(0, sql.length() - 3);
                        sql = sql + ")";
                        sql = sql + " and ";
                }
                if (!orderDate.equals("")) {
                        st = new StringTokenizer(orderDate, ",");
                        sql = sql + "(";
                        while (st.hasMoreTokens()) {
                                sql = sql + "orderDate like ? or ";
                                st.nextToken();
                        }
                        sql = sql.substring(0, sql.length() - 3);
                        sql = sql + ")";
                        sql = sql + " and ";
                }
                if (!totalPrice.equals("")) {
                        st = new StringTokenizer(totalPrice, ",");
                        sql = sql + "(";
```

```
                        switch (comboTotalPrice) {
                        case (0):
                                while (st.hasMoreTokens()) {
                                        sql = sql + "totalPrice = ? or ";
                                        st.nextToken();
                                }
                                break;
                        case (1):
                                while (st.hasMoreTokens()) {
                                        sql = sql + "Cast(totalPrice as char(15)) like ? or ";
                                        st.nextToken();
                                }
                                break;
                        case (2):
                                while (st.hasMoreTokens()) {
                                        sql = sql + "totalPrice >= ? or ";
                                        st.nextToken();
                                }
                                break;
                        case (3):
                                while (st.hasMoreTokens()) {
                                        sql = sql + "totalPrice <= ? or ";
                                        st.nextToken();
                                }
                                break;
                        }
                        sql = sql.substring(0, sql.length() - 3);
                        sql = sql + ")";
                        sql = sql + " and ";
                }
                sql = sql.substring(0, sql.length() - 4);
                try {
                        pstmt = con.prepareStatement(sql);
                        int idx = 1;
                        if (!orderNo.equals("")) {
                                st = new StringTokenizer(orderNo, ",");
                                while (st.hasMoreTokens()) {
                                        pstmt.setString(idx++, "%" + st.nextToken().trim() +
"%");
                                }
                        }
                        if (!ID.equals("")) {
                                if (!ID.toUpperCase().equals("ID")) {
                                        st = new StringTokenizer(ID, ",");
                                        while (st.hasMoreTokens()) {
```

```
                                        pstmt.setString(idx++, "%" +
st.nextToken().trim() + "%");
                                }
                        }
                }
                if (!productNo.equals("")) {
                        if (!productNo.equals("번호")) {
                                st = new StringTokenizer(productNo, ",");
                                switch (comboProductNo) {
                                case (1):
                                        while (st.hasMoreTokens()) {
                                                pstmt.setString(idx++, "%" +
st.nextToken().trim() + "%");
                                        }
                                        break;
                                default:
                                        while (st.hasMoreTokens()) {
                                                pstmt.setInt(idx++,
Integer.parseInt(st.nextToken().trim()));
                                        }
                                        break;
                                }
                        }
                }
                if (!orderAmount.equals("")) {
                        st = new StringTokenizer(orderAmount, ",");
                        switch (comboOrderAmount) {
                        case (1):
                                while (st.hasMoreTokens()) {
                                        pstmt.setString(idx++, "%" +
st.nextToken().trim() + "%");
                                }
                                break;
                        default:
                                while (st.hasMoreTokens()) {
                                        pstmt.setInt(idx++,
Integer.parseInt(st.nextToken().trim()));
                                }
                                break;
                        }
                }
                if (!orderDate.equals("")) {
                        st = new StringTokenizer(orderDate, ",");
                        while (st.hasMoreTokens()) {
                                pstmt.setString(idx++, "%" + st.nextToken().trim() +
```

```
"%");
                                        }
                                }
                                if (!totalPrice.equals("")) {
                                        st = new StringTokenizer(totalPrice, ",");
                                        switch (comboTotalPrice) {
                                        case (1):
                                                while (st.hasMoreTokens()) {
                                                        pstmt.setString(idx++, "%" +
st.nextToken().trim() + "%");
                                                }
                                                break;
                                        default:
                                                while (st.hasMoreTokens()) {
                                                        pstmt.setInt(idx++,
Integer.parseInt(st.nextToken().trim()));
                                                }
                                                break;
                                        }
                                }

                                rs = pstmt.executeQuery();
                                model.setRowCount(0);
                                int sum = 0;
                                while (rs.next()) {
                                        if (productNo.equals("번호")) {
                                                model.addRow(new Object[] { null, null, null,
rs.getInt("productNo"), rs.getInt("countPN"),
                                                                rs.getInt("sumPN") });
                                                sum += rs.getInt("sumPN");
                                        } else if (ID.toUpperCase().equals("ID")) {
                                                model.addRow(new Object[] { null, rs.getString("ID"),
null, null, rs.getInt("countID"),
                                                                rs.getInt("sumID") });
                                                sum += rs.getInt("sumID");
                                        } else {
                                                model.addRow(new Object[] { rs.getInt("orderNo"),
rs.getString("ID"), rs.getString("orderDate"),
                                                                rs.getInt("productNo"),
rs.getInt("orderAmount"), rs.getInt("totalPrice") });
                                                sum += rs.getInt("totalPrice");
                                        }
                                }

                                String sumResult = formatter.format(sum);
```

```
                        textSum.setText(sumResult);
                } catch (Exception ex) {
                        ex.printStackTrace();
                }
        }
```

7. ProductPanel.java: 상품화면 및 기능구현

7-2 상품 정보 갱신 기능

```
// 상품 정보를 입력하는 기능
        private static void insertProduct() {
                String sql = "INSERT INTO producttbl values(?,?,?,?,?,?,?)";
                try {
                        int idx = 1;
                        String date = textReDate.getText();
                        java.sql.Date sDate;
                        if(date.equals("")) {
                                sDate = null;
                        }else {
                                sDate = java.sql.Date.valueOf(date);
                        }
                        pstmt = conn.prepareStatement(sql);
                        pstmt.setInt(idx++, Integer.parseInt(textPNo.getText()));
                        pstmt.setString(idx++, textPName.getText());
                        pstmt.setString(idx++, textType.getText());
                        pstmt.setInt(idx++, Integer.parseInt(textCost.getText()));
                        pstmt.setInt(idx++, Integer.parseInt(textPrice.getText()));
                        pstmt.setInt(idx++, Integer.parseInt(textAmount.getText()));
                        pstmt.setDate(idx, sDate);
                        pstmt.executeUpdate();
                        model.setRowCount(0);
                        getTable();
                        if (pstmt != null)
                                pstmt.close();
                        String error = "상품 정보를 추가했습니다.";
                        JLabel lblError = new JLabel(error);
                        lblError.setFont(new Font("맑은 고딕", Font.PLAIN, 15));
                        JOptionPane.showMessageDialog(null, lblError,
"Successful", JOptionPane.PLAIN_MESSAGE);
                } catch (SQLException e1) {
                        String error = "상품 정보를 추가하지 못했습니다.";
                        JLabel lblError = new JLabel(error);
                        lblError.setFont(new Font("맑은 고딕", Font.PLAIN, 15));
                        JOptionPane.showMessageDialog(null, lblError, "Error",
JOptionPane.PLAIN_MESSAGE);
                } catch (IllegalArgumentException e1) {
                        String error = "날짜 형식을 맞춰서
입력하세요.(1999-12-31)";
                        JLabel lblError = new JLabel(error);
                        lblError.setFont(new Font("맑은 고딕", Font.PLAIN, 15));
                        JOptionPane.showMessageDialog(null, lblError, "Error",
JOptionPane.PLAIN_MESSAGE);
                }
        }
```

7-3 상품 정보 삭제 기능

```
// 상품 정보를 갱신하는 기능
        private static void updateProduct() {
```

```
            String sql = "UPDATE producttbl SET productNo = ?,
productName = ?, productType = ?, cost = ?, price = ?, amount = ?,
receivedDate = ? where productNo = ?";
            try {
                  int idx = 1;
                  String date = textReDate.getText();
                  java.sql.Date sDate;
                  if(date.equals("")) {
                        sDate = null;
                  }else {
                        sDate = java.sql.Date.valueOf(date);
                  }
                  pstmt = conn.prepareStatement(sql);
                  pstmt.setInt(idx++, Integer.parseInt(textPNo.getText()));
                  pstmt.setString(idx++, textPName.getText());
                  pstmt.setString(idx++, textType.getText());
                  pstmt.setInt(idx++, Integer.parseInt(textCost.getText()));
                  pstmt.setInt(idx++, Integer.parseInt(textPrice.getText()));
                  pstmt.setInt(idx++, Integer.parseInt(textAmount.getText()));
                  pstmt.setDate(idx++, sDate);
                  pstmt.setInt(idx++, (Integer) model.getValueAt(select, 0));
                  System.out.println(select);
                  pstmt.executeUpdate();
                  model.setRowCount(0);
                  getTable();
                  if (pstmt != null)
                        pstmt.close();
                  String error = "상품 정보를 갱신했습니다.";
                  JLabel lblError = new JLabel(error);
                  lblError.setFont(new Font("맑은 고딕", Font.PLAIN, 15));
                  JOptionPane.showMessageDialog(null, lblError,
"Successful", JOptionPane.PLAIN_MESSAGE);
            } catch (SQLException e1) {
                  String error = "상품 정보를 갱신하지 못했습니다.";
                  JLabel lblError = new JLabel(error);
                  lblError.setFont(new Font("맑은 고딕", Font.PLAIN, 15));
                  JOptionPane.showMessageDialog(null, lblError, "Error",
JOptionPane.PLAIN_MESSAGE);
            } catch (IllegalArgumentException e1) {
                  String error = "날짜 형식을 맞춰서
입력하세요.(1999-12-31)";
                  JLabel lblError = new JLabel(error);
                  lblError.setFont(new Font("맑은 고딕", Font.PLAIN, 15));
                  JOptionPane.showMessageDialog(null, lblError, "Error",
JOptionPane.PLAIN_MESSAGE);
            }
      }
```

7-4 상품 정보 검색 기능

```
// 상품 정보를 검색하는 기능
      private static void searchProduct(String productNo, String productName, String
productType, String cost,
                  String price, String amount, String receivedDate, int a, int b, int c)
{
            // 0 단일 1 포함 2 이상 3 이하
            String sql = "SELECT * FROM producttbl WHERE ";
//          String sql = "select * from producttbl where Cast(price as char) like ?";
```

```
		if (!productNo.equals("")) {
			st = new StringTokenizer(productNo, ",");
			sql = sql + "(";
			while (st.hasMoreTokens()) {
				sql = sql + "Cast(productNo as char(10)) like ? or ";
				st.nextToken();
			}
			sql = sql.substring(0, sql.length() - 3);
			sql = sql + ")";
			sql = sql + " and ";
		}
		if (!productName.equals("")) {
			st = new StringTokenizer(productName, ",");
			sql = sql + "(";
			while (st.hasMoreTokens()) {
				sql = sql + "productName like ? or ";
				st.nextToken();
			}
			sql = sql.substring(0, sql.length() - 3);
			sql = sql + ")";
			sql = sql + " and ";
		}
		if (!productType.equals("")) {
			st = new StringTokenizer(productType, ",");
			sql = sql + "(";
			while (st.hasMoreTokens()) {
				sql = sql + "productType like ? or ";
				st.nextToken();
			}
			sql = sql.substring(0, sql.length() - 3);
			sql = sql + ")";
			sql = sql + " and ";
		}
		if (!cost.equals("")) {
			switch (a) {
			case 0:// 단일
				st = new StringTokenizer(cost, ",");
				sql = sql + "(";
				while (st.hasMoreTokens()) {
					sql = sql + "cost = ? or ";
					st.nextToken();
				}
				sql = sql.substring(0, sql.length() - 3);
				sql = sql + ")";
				sql = sql + " and ";
```

```
                        break;
                case 1:// 포함
                        st = new StringTokenizer(cost, ",");
                        sql = sql + "(";
                        while (st.hasMoreTokens()) {
                                sql = sql + "Cast(cost as char(10)) like ? or ";
                                st.nextToken();
                        }
                        sql = sql.substring(0, sql.length() - 3);
                        sql = sql + ")";
                        sql = sql + " and ";
                        break;
                case 2:// 이상
                        st = new StringTokenizer(cost, ",");
                        sql = sql + "(";
                        while (st.hasMoreTokens()) {
                                sql = sql + "cost >= ? or ";
                                st.nextToken();
                        }
                        sql = sql.substring(0, sql.length() - 3);
                        sql = sql + ")";
                        sql = sql + " and ";
                        break;
                default:// 이하
                        st = new StringTokenizer(cost, ",");
                        sql = sql + "(";
                        while (st.hasMoreTokens()) {
                                sql = sql + "cost <= ? or ";
                                st.nextToken();
                        }
                        sql = sql.substring(0, sql.length() - 3);
                        sql = sql + ")";
                        sql = sql + " and ";
                        break;
                }
        }
        if (!price.equals("")) {
                switch (b) {
                case 0:// 단일
                        st = new StringTokenizer(price, ",");
                        sql = sql + "(";
                        while (st.hasMoreTokens()) {
                                sql = sql + "price = ? or ";
                                st.nextToken();
                        }
```

```
                        sql = sql.substring(0, sql.length() - 3);
                        sql = sql + ")";
                        sql = sql + " and ";
                        break;
                case 1:// 포함
                        st = new StringTokenizer(price, ",");
                        sql = sql + "(";
                        while (st.hasMoreTokens()) {
                                sql = sql + "Cast(price as char(10)) like ? or ";
                                st.nextToken();
                        }
                        sql = sql.substring(0, sql.length() - 3);
                        sql = sql + ")";
                        sql = sql + " and ";
                        break;
                case 2:// 이상
                        st = new StringTokenizer(price, ",");
                        sql = sql + "(";
                        while (st.hasMoreTokens()) {
                                sql = sql + "price >= ? or ";
                                st.nextToken();
                        }
                        sql = sql.substring(0, sql.length() - 3);
                        sql = sql + ")";
                        sql = sql + " and ";
                        break;
                default:// 이하
                        st = new StringTokenizer(price, ",");
                        sql = sql + "(";
                        while (st.hasMoreTokens()) {
                                sql = sql + "price <= ? or ";
                                st.nextToken();
                        }
                        sql = sql.substring(0, sql.length() - 3);
                        sql = sql + ")";
                        sql = sql + " and ";
                        break;
                }
        }
        if (!amount.equals("")) {
                switch (c) {
                case 0:// 단일
                        st = new StringTokenizer(amount, ",");
                        sql = sql + "(";
                        while (st.hasMoreTokens()) {
```

```
                                sql = sql + "amount = ? or ";
                                st.nextToken();
                        }
                        sql = sql.substring(0, sql.length() - 3);
                        sql = sql + ")";
                        sql = sql + " and ";
                        break;
                case 1:// 포함
                        st = new StringTokenizer(amount, ",");
                        sql = sql + "(";
                        while (st.hasMoreTokens()) {
                                sql = sql + "Cast(amount as char(10)) like ? or ";
                                st.nextToken();
                        }
                        sql = sql.substring(0, sql.length() - 3);
                        sql = sql + ")";
                        sql = sql + " and ";
                        break;
                case 2:// 이상
                        st = new StringTokenizer(amount, ",");
                        sql = sql + "(";
                        while (st.hasMoreTokens()) {
                                sql = sql + "amount >= ? or ";
                                st.nextToken();
                        }
                        sql = sql.substring(0, sql.length() - 3);
                        sql = sql + ")";
                        sql = sql + " and ";
                        break;
                default:// 이하
                        st = new StringTokenizer(amount, ",");
                        sql = sql + "(";
                        while (st.hasMoreTokens()) {
                                sql = sql + "amount <= ? or ";
                                st.nextToken();
                        }
                        sql = sql.substring(0, sql.length() - 3);
                        sql = sql + ")";
                        sql = sql + " and ";
                        break;
                }
        }
        if (!receivedDate.equals("")) {
                st = new StringTokenizer(receivedDate, ",");
                sql = sql + "(";
```

```
                        while (st.hasMoreTokens()) {
                                sql = sql + "receivedDate like ? or ";
                                st.nextToken();
                        }
                        sql = sql.substring(0, sql.length() - 3);
                        sql = sql + ")";
                        sql = sql + " and ";
                }
                sql = sql.substring(0, sql.length() - 4);
                System.out.println(sql);
                try {
                        int idx = 1;
//                      String sql = "select * from producttbl where Cast(price as char) like
?";
                        pstmt = conn.prepareStatement(sql);
                        if (!productNo.equals("")) {
                                st = new StringTokenizer(productNo, ",");
                                while (st.hasMoreTokens()) {
                                        pstmt.setString(idx++, "%" + st.nextToken().trim() +
"%");
                                }
                        }
                        if (!productName.equals("")) {
                                st = new StringTokenizer(productName, ",");
                                while (st.hasMoreTokens()) {
                                        pstmt.setString(idx++, "%" + st.nextToken().trim() +
"%");
                                }
                        }
                        if (!productType.equals("")) {
                                st = new StringTokenizer(productType, ",");
                                while (st.hasMoreTokens()) {
                                        pstmt.setString(idx++, "%" + st.nextToken().trim() +
"%");
                                }
                        }
                        if (!cost.equals("")) {
                                switch (a) {
                                case 1: // 포함
                                        st = new StringTokenizer(cost, ",");
                                        while (st.hasMoreTokens()) {
                                                pstmt.setString(idx++, "%" +
st.nextToken().trim() + "%");
                                        }
                                        break;
```

```
                                default: // 단일, 이상, 이하
                                        st = new StringTokenizer(cost, ",");
                                        while (st.hasMoreTokens()) {
                                                pstmt.setInt(idx++,
Integer.parseInt(st.nextToken()));
                                        }
                                        break;
                                }
                        }
                        if (!price.equals("")) {
                                switch (b) {
                                case 1: // 포함
                                        st = new StringTokenizer(price, ",");
                                        while (st.hasMoreTokens()) {
                                                pstmt.setString(idx++,          "%"          +
st.nextToken().trim() + "%");
                                        }
                                        break;
                                default: // 단일, 이상, 이하
                                        st = new StringTokenizer(price, ",");
                                        while (st.hasMoreTokens()) {
                                                pstmt.setInt(idx++,
Integer.parseInt(st.nextToken()));
                                        }
                                        break;
                                }
                        }
                        if (!amount.equals("")) {
                                switch (c) {
                                case 1: // 포함
                                        st = new StringTokenizer(amount, ",");
                                        while (st.hasMoreTokens()) {
                                                pstmt.setString(idx++,          "%"          +
st.nextToken().trim() + "%");
                                        }
                                        break;
                                default: // 단일, 이상, 이하
                                        st = new StringTokenizer(amount, ",");
                                        while (st.hasMoreTokens()) {
                                                pstmt.setInt(idx++,
Integer.parseInt(st.nextToken()));
                                        }
                                        break;
                                }
                        }
```

```
                        if (!receivedDate.equals("")) {
                                st = new StringTokenizer(receivedDate, ",");
                                while (st.hasMoreTokens()) {
                                        pstmt.setString(idx++, "%" + st.nextToken().trim() +
"%");
                                }
                        }
//                      pstmt.setString(1, "%5%");
                        rs = pstmt.executeQuery();
                        model.setRowCount(0);
                        while (rs.next()) {
                                model.addRow(new Object[] { rs.getInt("productNo"),
rs.getString("productName"),
                                                rs.getString("producttype"), rs.getInt("cost"),
rs.getInt("price"), rs.getInt("amount"),
                                                rs.getString("receivedDate") });
                        }
                        if (rs != null)
                                rs.close();
                        if (pstmt != null)
                                pstmt.close();
                } catch (SQLException e1) {
                        e1.printStackTrace();
                }
        }
```

7-6 상품 정보 중복 제거 검색 기능

```
// 상품 정보에서 상품명과 분류를 중복없이 검색하는 기능
        private static void distinctSearch(String sel) {
                model.setRowCount(0);
                String sql = "";
                if (sel.equals("상품명"))
                        sql = "SELECT DISTINCT productName FROM producttbl";
                else if (sel.equals("분류"))
                        sql = "SELECT DISTINCT productType FROM producttbl";
                try {
                        pstmt = conn.prepareStatement(sql);
                        rs = pstmt.executeQuery();
                        while (rs.next()) {
                                if (sel.trim().equals("상품명"))
                                        model.addRow(new Object[] { null,
rs.getString("productName"), null, null, null, null, null });
                                else if (sel.trim().equals("분류"))
                                        model.addRow(new Object[] { null, null,
rs.getString("producttype"), null, null, null, null });
```

```
                        }
                        if (rs != null)
                                rs.close();
                        if (pstmt != null)
                                pstmt.close();
                } catch (SQLException e) {
                        // TODO Auto-generated catch block
                        e.printStackTrace();
                }
        }
```

8. RightNumberRenderer.java : 숫자 자리수 구분

```
public class RightNumberRenderer extends DefaultTableCellRenderer {

        private final DecimalFormat formatter = new DecimalFormat("###,###,###,###");

        public final Component getTableCellRendererComponent(JTable table, Object value,
boolean isSelected,
                        boolean hasFocus, int row, int column) {
                if (value == null) {
                        value = 0;
                }
                setHorizontalAlignment(JLabel.RIGHT);
                value = formatter.format((Number) value);
                final Component result = super.getTableCellRendererComponent(table, value,
isSelected, hasFocus, row, column);

                return result;
        }
}
```

9. RoundedButton.java : 버튼 모서리 둥글게 처리

```
class RoundedButton extends JButton {
        public RoundedButton() {
                super();
                decorate();
        }
        public RoundedButton(String text) {
                super(text);
                decorate();
        }
        public RoundedButton(Action action) {
                super(action);
                decorate();
        }
```

```
	public RoundedButton(Icon icon) {
		super(icon);
		decorate();
	}
	public RoundedButton(String text, Icon icon) {
		super(text, icon);
		decorate();
	}
	protected void decorate() {
		setBorderPainted(false);
		setOpaque(false);
	}
	@Override
	protected void paintComponent(Graphics g) {
		int width = getWidth();
		int height = getHeight();
		Graphics2D graphics = (Graphics2D) g;
		graphics.setRenderingHint(RenderingHints.KEY_ANTIALIASING,
RenderingHints.VALUE_ANTIALIAS_ON);
		if (getModel().isArmed()) {
			graphics.setColor(getBackground().darker());
		} else if (getModel().isRollover()) {
			graphics.setColor(getBackground().brighter());
		} else {
			graphics.setColor(getBackground());
		}
		graphics.fillRoundRect(0, 0, width, height, 10, 10);
		FontMetrics fontMetrics = graphics.getFontMetrics();
		Rectangle stringBounds = fontMetrics.getStringBounds(this.getText(),
graphics).getBounds();
		int textX = (width - stringBounds.width) / 2;
		int textY = (height - stringBounds.height) / 2 + fontMetrics.getAscent();
		graphics.setColor(getForeground());
		graphics.setFont(getFont());
		graphics.drawString(getText(), textX, textY);
		graphics.dispose();
		super.paintComponent(g);
	}
}
```